IDE Insights – Part 1

Component Code Markers

Date: July 17, 2003

Document responsible: John van Houten

The information in this document pertains to Visual DataFlex version 9.1. The information is provided “as-is”. The author accepts no responsibility for the accuracy of the presented information.

The VDF 9.1 IDE has much better support for integrating custom wizard applications for building components. If you are interested in building a wizard or any other code generator, then the following information should be useful…

The IDE’s Visual Designer maintains a set of special markers inside the source code of each component file. These markers are parsed whenever a component is opened, and re-generated when the component is written.

Here is a brief description of each special marker...

```
//AB-StoreTopStart
//AB-StoreTopEnd
```

These markers mark the block of custom code between an object declaration and the IDE generated property settings for that object.

```
//AB-StoreStart
//AB-StoreEnd
```

These markers mark the block of custom code between the IDE generated property settings of an object (and any nested IDE generated objects) and the End_Object statement

```
//AB-IgnoreStart
//AB-IgnoreEnd
```

These markers denote a block of code that is not read in when the component is opened in the IDE. The IDE essentially discards any code that lies between these markers when the component is opened. Typically, when the component is saved, the IDE will regenerate a new section of code between these markers.

```
//AB-RegisterInterfaceStart
//AB-RegisterInterfaceEnd
```

These markers are used by Web Object components that would be compiled into a Web Server Application. They surround the list of declarations that register the external interface a Web Object. For example a Web Browser Object's registered interface would be declared here. In the case of a WBO the register interface section would have to appear inside the cWebBusinessProcess object.

The only code that should exist between these markers are "Send RegisterInterface" statements. All other code is discarded by the IDE when the component is opened. A Send RegisterInterface

statement should be written as follows (it is better if the statement can be written all on one line, I have broken it here for readability).

```
Send RegisterInterface msg_{MethodName} ;
                       "msg_{MethodName}" ;
                       "{ParamType1..n} {ParamName1..n}" ;
                       "{Description}"
```

```
//AB-MenuAutogen
```

This marker can be used in an MDI Application. The marker would appear inside the Panel object of the MDI Application component. The marker sets a special flag in the IDE to tell it to automatically generate the menu bar each time the component is saved. The Menu Bar consists of standard File, Navigate, Windows and Help Menus. In addition a View and Report menu are generated containing options for opening each View and Report component that belongs to the application.

This marker is controlled by the "Auto-Generate Menus" checkbox in the MDI Panel Properties dialog. It is hoped that this marker will become obsolete once the IDE supports better menu modeling.

```
//AB-MenuPackage
//AB-End
```

This marker is used in the same types of components as //AB-MenuAutogen. This marker and //AB-MenuAutogen are mutually exclusive. In-between this marker and the //AB-End marker, the IDE maintains the name of a package file that contains the component's menubar object declarations.

This marker is controlled by the "Menu Package" textbox in the MDI Panel Properties dialog. It is hoped that this marker will become obsolete once the IDE supports better menu modeling.

```
//AB-ToolbarPackage
//AB-End
```

This marker serves the same purpose for an MDI Application's toolbar as //AB-MenuPackage does for its menu. It is also hoped that this marker will become obsolete once the IDE supports better toolbar modeling.

```
//AB-StatusBarPackage
//AB-End
```

This marker serves the same purpose for an MDI Application's statusbar as //AB-MenuPackage does for its menu. It is also hoped that this marker will become obsolete once the IDE supports better statusbar modeling.

```
//AB-ViewStart
//AB-ViewEnd
```

These markers appear inside any object that is designed to support "component nesting". For example, an AppClientArea object supports nesting of View, Lookup, Dialog, Report and Business Process components. A cWebApp object supports nesting of BusinessProcess and Web Object components. Between these markers, the IDE maintains a list of Use statements for each nested component.

```
//AB-DDOStart
```

```
//AB-DDOEnd
```

These markers appear inside any object that is designed to support data dictionary object structure nesting. For example, a dbView, a BusinessProcess, or a dbTabView. Between these markers, the IDE maintains a structure of data dictionary object declarations.

```
//AB/
```

This is a very useful and flexible marker. It is used to store a property (or even an object declaration) which is executed in the IDE's Visual Designer but is saved as a comment (not executed at runtime). This is how the IDE maintains Size and Location properties for non-visual components such as cWebBusinessProcess, cApplication, Array, or SaveAsDialog. If you want a property to be modeled in this way, then you must declare the property in the class's meta-data as “Pseudo”. For example...

```
Property No_Visible Pseudo Complex Location 0 0
```

There are some others markers however they have been obsolete since VDF 8.2. Here is a list of those obsolete markers:

```
//AB-PanelStoreStart
//AB-PanelStoreEnd
//AB-PanelStoreTopStart
//AB-PanelStoreTopEnd

//AB-ClientStoreStart
//AB-ClientStoreEnd
//AB-ClientStoreTopStart
//AB-ClientStoreTopEnd
```